

6. 用户（超级用户，系统用户和普通用户）和组

id

id user02

ls -l 文件

ls -dl 文件夹

ps -au 第一列显示用户名

以上命令的输出按名称显示用户，但操作系统内部利用 UID 来跟踪用户。用户名到 UID 的映射在帐户信息数据库中定义。默认情况下，系统使用 **/etc/passwd** 文件存储有关本地用户的信息。

/etc/passwd 文件的每一行都包含了有关某个用户的信息。它分为七个以冒号分隔的字段。以下是 **/etc/passwd** 中某一行的示例：

```
①user01:②x:③1000:④1000:⑤User One:⑥/home/user01:⑦/bin/bash
```

- ① 该用户 (**user01**) 的用户名。
- ② 用户的密码曾以加密格式存储在此处。现在它已移至 **/etc/shadow** 文件，稍后我们将对此进行介绍。**[该字段始终应为 x]**
- ③ 该用户帐户的 UID 号 (**1000**)。
- ④ 该用户帐户的主要组的 GID 号 (**1000**)。本节稍后将对组进行讨论。
- ⑤ 该用户的真实姓名 (**User One**)。
- ⑥ 该用户的主目录 (**/home/user01**)。这是 shell 启动时的初始工作目录，其中包含有用户数据和配置设置。
- ⑦ 该用户的默认 shell 程序，会在登录时运行 (**/bin/bash**)。对于普通用户，这通常是提供用户命令行提示符的程序。如果系统用户不允许进行交互式登录，该用户可能会使用 **/sbin/nologin**。

/etc/group 文件的每一行都包含了有关某个组的信息。每个组条目被分为四个以冒号分隔的字段。以下是 **/etc/group** 中某一行的示例：

```
①group01:②x:③10000:④user01,user02,user03
```

- ① 该组的组名称 (**group01**)。
- ② 过时的组密码字段。该字段始终应为 x。
- ③ 该组的 GID 号 (**10000**)。

地址：广州市天河区棠安路 188 号乐天大厦：

电话：020-38289118

RH124-L8.0-zh_cn-1-20190531

网址：www.togogo.net

Your IT Value



第 6 | 管理本地用户和组

- ④ 属于作为补充组的该组成员的用户列表 (**user01**、**user02**、**user03**)。本节稍后将对主要组（或默认组）和补充组进行讨论。

主要组和补充组

每个用户有且只有一个**主要组**。对于本地用户而言，这是按 **/etc/passwd** 文件中的 GID 号列出的组。默认情况下，这是拥有用户创建的新文件的组。

通常，在创建新的普通用户时，会创建一个与该用户同名的新组。该组将用作新用户的主要组，而该用户是这一用户专用组的唯一成员。事实证明，这有助于简化文件权限的管理。本课程的后续部分将对此进行讨论。

用户也可以有补充组。**补充组的成员资格由 /etc/group 文件确定**，根据所在的组是否具有访问权限，将授予用户对文件的访问权限。具有访问权限的组是用户的主要组还是补充组无关紧要。

例如，如果用户 **user01** 有一个主要组 **user01** 以及两个补充组 **wheel** 和 **webadmin**，那么该用户就可以读取其中任何一个组可读的文件。

该 **id** 命令还可用于查找用户的组成员身份。

```
[user03@host ~]$ id  
uid=1003(user03) gid=1003(user03) groups=1003(user03),10(wheel),10000(group01)  
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

在上面的示例中，**user03** 将组 **user03** 作为自己的主要组 (**gid**)。**groups** 项目列出了该用户的所有组。除了主要组 **user03** 外，用户还有组 **wheel** 和 **group01** 作为补充组。



参考文献

id(1)、**passwd(5)** 和 **group(5)** man page

info libc (GNU C 库参考书册)

· 第 30 节：用户和组

(请注意，必须安装 glibc-devel 软件包，此 info 节点才可用。)

获取超级用户访问权限

```
Password:  
[root@host ~]#
```

命令 **su** 将启动非登录 shell，而命令 **su -**（带有短划线选项）会启动登录 shell。两个命令的主要区别在于，**su -** 会将 shell 环境设置为如同以该用户身份重新登录一样，而 **su** 仅以该用户身份启动 shell，但使用的是原始用户的环境设置。

在大多数情况下，管理员应该运行 **su -** 以获得包含目标用户常规环境设置的 shell。有关详细信息，请参阅 **bash(1)** man page。



注意

su 命令最常用于获得以另一个用户身份（通常是 **root**）运行的命令行界面（shell 提示符）。但是，如果结合 **-c** 选项，该命令的作用将与 Windows 实用程序 **runas** 一样，能够以另一个用户身份运行任意程序。运行 **info su** 来查看更多详情。

与 **su** 不同，**sudo** 通常要求用户输入其自己的密码以进行身份验证，而不是输入他们正尝试访问的用户帐户的密码。也就是说，用户使用 **sudo** 以 **root** 运行命令时，不需要知道 **root** 密码。相反，他们将使用自己的密码来验证访问权限。

此外，**sudo** 可以配置为允许特定用户像某个其他用户一样运行任何命令，或仅允许以该用户身份运行部分命令。

例如，如果 **sudo** 已配置为允许 **user01** 用户以 **root** 身份运行命令 **usermod**，那么 **user01** 就可运行以下命令来锁定或解锁用户帐户：

通过 Sudo 获取交互式 Root Shell

如果系统上的非管理员用户帐户能够使用 **sudo** 来运行 **su** 命令，则可以从该帐户运行 **sudo su -** 来获取 **root** 用户的交互式 shell。这是因为 **sudo** 将以 **root** 用户身份运行 **su -**，而 **root** 用户无需输入密码即可使用 **su**。

通过 **sudo** 访问 **root** 帐户的另一种方式是使用 **sudo -i** 命令。这将切换到 **root** 帐户并运行该用户的默认 shell（通常为 **bash**）及关联的 shell 登录脚本。如果您只想运行 shell，可使用 **sudo -s** 命令。

例如，管理员可以获取 AWS EC2 实例上 **root** 用户的交互式 shell，方法是使用 SSH 公钥身份验证作为普通用户 **ec2-user** 登录，再运行 **sudo -i** 来获取 **root** 用户的 shell。

```
[ec2-user@host ~]$ sudo -i  
[sudo] password for ec2-user:  
[root@host ~]#
```

sudo su - 命令与 **sudo -i** 的行为不完全相同。相关内容会在本节的末尾部分进行简要讨论。

管理本地用户账户

- **usermod --help** 命令显示可用于修改帐户的基本选项。一些常见选项包括：

USERMOD 选项:	用法
-c, --comment COMMENT	将用户的真实姓名添加到注释字段。
-g, --gid GROUP	为用户帐户指定主要组。
-G, --groups GROUPS	为用户帐户指定补充组的逗号分隔列表。
-a, --append	利用 -G 选项将补充组添加到用户当前的组成员集合中，而不是将补充组集合替换为新的集合。
-d, --home HOME_DIR	为用户帐户指定特定的主目录。
-m, --move-home	将用户的主目录移到新的位置。必须与 -d 选项搭配使用。
-s, --shell SHELL	为用户帐户指定特定的登录 shell。
-L, --lock	锁定用户帐户。
-U, --unlock	解锁用户帐户。

- **userdel -r username** 命令从 **/etc/passwd** 中删除 **username** 的详细信息，同时删除用户的主目录。

- UID 0 始终分配至超级用户帐户 **root**。
- UID 1-200 是一系列“系统用户”，由红帽静态分配给系统进程。
- UID 201-999 是一系列“系统用户”，供文件系统中没有自己的文件的系统进程使用。通常在安装需要它们的软件时，从可用池中动态分配它们。程序以这些“无特权”系统用户身份运行，以便限制它们仅访问正常运行所需的资源。
- UID 1000+ 是可供分配给普通用户的范围。

管理本地组账户

从命令行创建组

· **groupadd** 命令用于创建组。不带选项时, **groupadd** 命令会在创建组时使用 **/etc/login.defs** 文件中指定范围内的下一个可用 GID。

· **-g** 选项指定要使用的组的特定 GID。

```
[user01@host ~]$ sudo groupadd -g 10000 group01
[user01@host ~]$ tail /etc/group
...output omitted...
group01:x:10000:
```

注意

由于用户私有组 (GID 1000 以上) 是自动创建的, 因此通常建议预备一系列 GID 待用于补充组。较高的范围可以避免与系统组 (GID 0-999) 产生冲突。

· **-r** 选项使用 **/etc/login.defs** 文件中所列有效系统 GID 范围内的 GID 创建系统组。**/etc/login.defs** 中的 **SYS_GID_MIN** 和 **SYS_GID_MAX** 配置项定义系统 GID 的范围。

```
[user01@host ~]$ sudo groupadd -r group02
[user01@host ~]$ tail /etc/group
...output omitted...
group01:x:10000:
group02:x:988:
```

从命令行修改现有的组

· **groupmod** 命令可更改现有组的属性。**-n** 选项指定组的新名称。

```
[user01@host ~]$ sudo groupmod -n group0022 group02
[user01@host ~]$ tail /etc/group
...output omitted...
group0022:x:988:
```

注意组名称已从 group02 更新为 group0022。

从命令行删除组

- **groupdel** 命令可删除组。

```
[user01@host ~]$ sudo groupdel group0022
```

注意

如果组是任何现有用户的主要组，则您无法删除它。与 **userdel** 一样，请检查所有文件系统，确保系统上不遗留由该组拥有的任何文件。

从命令行更改组成员资格

- 组成员资格通过用户管理进行控制。使用 **usermod -g** 命令来更改用户的主要组。

```
[user01@host ~]$ id user02  
uid=1006(user02) gid=1008(user02) groups=1008(user02)  
[user01@host ~]$ sudo usermod -g group01 user02  
[user01@host ~]$ id user02  
uid=1006(user02) gid=10000(group01) groups=10000(group01)
```

- 使用 **usermod -aG** 命令，将用户添加到某一补充组。

```
[user01@host ~]$ id user03  
uid=1007(user03) gid=1009(user03) groups=1009(user03)  
[user01@host ~]$ sudo usermod -aG group01 user03  
[user01@host ~]$ id user03  
uid=1007(user03) gid=1009(user03) groups=1009(user03),10000(group01)
```

管理用户密码

阴影密码和密码策略

加密的密码一度存储在全局可读的 `/etc/passwd` 文件中。这曾被认为具有合理的安全性，直到对加密密码的字典式攻击变得常见。那时，加密的密码被移动到只有 `root` 用户才能读取的独立 `/etc/shadow` 文件中。这种新文件也允许实施密码期限和到期功能。

与 `/etc/passwd` 相似，每个用户在 `/etc/shadow` 文件中都有对应的一行。下方显示了来自 `/etc/shadow` 的示例行，以及其九个冒号分隔字段。

```
$ 1user03:2$6$CSsX...output omitted...:317933:40:599999:67:72:818113:9
```

- ① 此密码所属帐户的用户名。
- ② 此用户的加密密码。加密密码的格式将在本节后面讨论。
- ③ 上次更改密码的日期。其设置值为自 1970 年 1 月 1 日起的天数，并按 UTC 时区计算。
- ④ 自用户上次更改密码以来到可以再次更改之前必须经过的最短天数。
- ⑤ 在密码过期之前不进行密码更改可以经过的最长天数。空字段表示它不会根据上次更改以来的时间失效。
- ⑥ 警告期。当用户在截止日期之前登录达到该天数时，会收到有关密码过期的警告。
- ⑦ 非活动期。一旦密码过期，在这些天内仍可以接受登录。过了这一时期后，帐户将被锁定。
- ⑧ 密码过期日。其设置值为自 1970 年 1 月 1 日起的天数，并按 UTC 时区计算。空字段表示它不在特定的日期失效。
- ⑨ 最后一个字段通常为空，预留给未来使用。

加密密码的格式

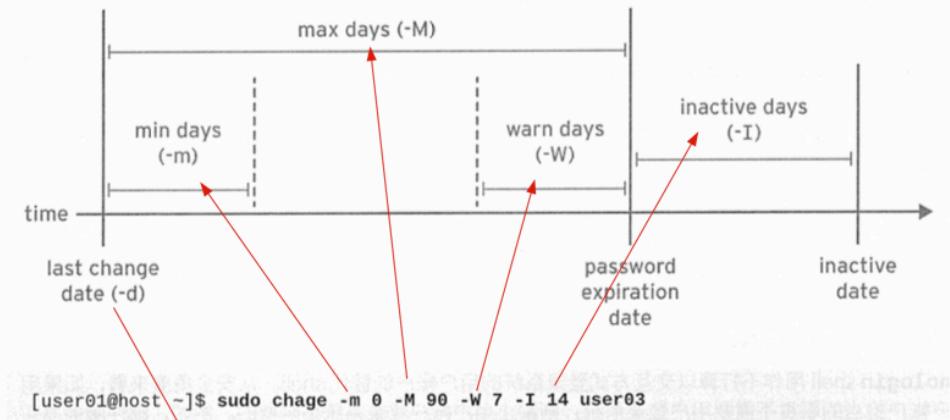
加密密码字段中存储了三段信息：所用的哈希算法、salt 及加密哈希值。每段信息由 `$` 符号分隔。

```
$ 16$2CSsXcYG1L/4ZfHr/$2W6evvJahUfzfHpc9X.45Jc6H30E...output omitted...
```

- ① 此密码所用的哈希算法。数字 **6** 表示 **SHA-512 哈希算法**，这是红帽企业 Linux 8 中的默认算法。**1** 表示 MD5 哈希算法，**5** 表示 SHA-256 哈希算法。
- ② 用于加密密码的 salt。这原先是随机选取的。
- ③ 用户密码的加密哈希值。**salt 和未加密密码组合并加密**，生成加密的密码哈希。

使用 salt 可以防止两个密码相同的用户在 `/etc/shadow` 文件中拥有相同的条目。例如，即使 `user01` 和 `user02` 都将 `redhat` 用作其密码，如果 salt 不同，那么它们在 `/etc/shadow` 中的加密密码也是不同的。

下图显示了相关的密码期限参数，可以通过 **chage** 命令对其调整来实施密码期限策略。



前面的 **chage** 命令使用了 **-m**、**-M**、**-W** 和 **-I** 选项，它们分别用于设置用户密码的最短期限、最长期限、警告周期和失效期限。

chage -d 0 user03 命令强制 user03 用户在下一次登录时更新其密码。

chage -l user03 命令显示 user03 的密码期限详情。

chage -E 2019-08-05 user03 命令使 user03 用户的帐户于 2019-08-05 到期 (YYYY-MM-DD 格式)。



注意

date 命令可用于计算未来的日期。**-u** 选项报告 UTC 时间。

```
[user01@host ~]$ date -d "+45 days" -u  
Thu May 23 17:01:20 UTC 2019
```

编辑 **/etc/login.defs** 文件中的密码期限配置项，以设置默认的密码期限策略。PASS_MAX_DAYS 设置密码的默认最长期限。PASS_MIN_DAYS 设置密码的默认最短期限。PASS_WARN_AGE 设置密码的默认警告周期。默认密码期限策略的任何更改都仅对新用户有效。现有用户将继续使用旧密码期限设置，而非新密码设置。

限制访问

您可以使用 **chage** 命令来设置帐户到期日期。到了该日期时，用户无法以交互方式登录系统。**usermod** 命令可以通过 **-L** 选项锁定帐户。

限制访问

您可以使用 **chage** 命令来设置帐户到期日期。到了该日期时，用户无法以交互方式登录系统。**usermod** 命令可以通过 **-L** 选项锁定帐户。

要直接登录该系统。

这种情况的常用解决方案是将用户的登录 shell 设为 **/sbin/nologin**。如果用户试图直接登录系统，**nologin** shell 将关闭该连接。

```
[user01@host ~]$ usermod -s /sbin/nologin user03
[user01@host ~]$ su - user03
Last login: Wed Feb  6 17:03:06 IST 2019 on pts/0
This account is currently not available.
```

7. Linux 文件系统权限

权限	对文件的影响	对目录的影响
r (读取)	可以读取文件内容。	可以列出目录的内容（文件名）。
w (写入)	可以更改文件内容。	可以创建或删除目录中的任一文件。
x (执行)	可以作为命令执行文件。	目录可以成为当前工作目录。（您可以 cd 它，但还需要读取权限才能列出里面的文件。）

x 进入目录 r 列出目录下文件

从命令行管理文件系统权限 chmod

通过符号法更改权限

`chmod WhoWhatWhich file|directory`

- Who 是指 u、g、o、a（代表用户、组、其他、全部）
- What 是指 +、-、=（代表添加、删除、精确设置）
- Which 是指 r、w、x（代表读取、写入、执行）

更改文件权限的符号法使用字母代表不同的权限组：**u** 表示用户，**g** 表示组，**o** 表示其他，**a** 表示全部。

使用符号法时，不需要设置一组全新的权限。取而代之，您可以更改现有的一个或多个权限。使用 **+** 或 **-** 来分别添加或删除权限，或者使用 **=** 来替换一组权限的整个集合。

权限自身由单个字母来表示：**r** 表示读取，**w** 表示写入，**x** 表示执行。在使用 **chmod** 通过符号法来更改权限时，仅当文件是目录或者已经为用户、组或其他人设置了执行权限时，使用大写的 **X** 作为权限标志才会添加执行权限。

注意

chmod 命令支持 **-R** 选项以递归方式对整个目录树中的文件设置权限。在使用 **-R** 选项时，使用 **X** 选项以符号形式设置权限会非常有用。这将能够对目录设置执行（搜索）权限，以便在不更改大部分文件权限的情况下，访问这些目录的内容。不过，使用 **X** 选项时要谨慎，因为如果某个文件设置有任何执行权限，则 **X** 也将会对该文件设置指定的执行权限。例如，以下命令会以递归方式为组所有者设置对 **demodir** 及其所有子代的读、写访问权限，但将仅向已为用户、组或其他人设置了执行权限的目录和文件应用组执行权限。

```
[root@host opt]# chmod -R g+rwx demodir
```

更改文件和目录的用户或组所有权

更改文件和目录的用户或组所有权

新建的文件由创建该文件的用户所有。默认情况下，新文件的组所有权为创建该文件的主要用户组。在红帽企业 Linux 中，用户的主要组通常为仅有该用户作为成员的私有组。要根据组成员资格授予对文件的访问权限，可能需要更改拥有该文件的组。

只有 **root** 用户可以更改拥有文件的用户。不过，组所有权可以由 **root** 用户或文件的所有者来设置。**root** 用户可将文件所有权授予任何组，而普通用户仅可将文件所有权授予他们所属的组。

使用 **chown**（更改所有者）命令可更改文件所有权。例如，要将文件 **test_file** 的所有权授予 **student** 用户，可使用以下命令：

```
[root@host ~]# chown student test_file
```

chown 可与 **-R** 选项配合使用，以递归更改整个目录树的所有权。以下命令可将 **test_dir** 的所有权以及此目录树中的所有文件和子目录授予给 **student**：

```
[root@host ~]# chown -R student test_dir
```

chown 命令也可用于更改文件的组所有权，只需在组名称之前加上冒号 (**:**)。例如，以下命令将 **test_dir** 组更改为 **admins**：

```
[root@host ~]# chown :admins test_dir
```

chown 命令也可用于同时更改所有者和组，此时可使用语法 **owner:group**。例如，要将 **test_dir** 的所有权更改为 **visitor**，同时将组更改为 **guests**，可使用以下命令：

```
[root@host ~]# chown visitor:guests test_dir
```

有些用户并不使用 **chown**，而是使用 **chgrp** 命令来更改组所有权。该命令的作用与 **chown** 类似，不同之处在于它仅用于更改组所有权，而且不需要组名前的冒号 (**:**)。



重要

您可能会遇到 **chown** 命令使用替代语法的示例，该语法用句点而不是冒号分隔所有者和组：

```
[root@host ~]# chown owner.group filename
```

您不应该使用此语法。应始终使用冒号。

句点是用户名中的有效字符，但冒号不是。如果系统上存在用户 **enoch.root**、用户 **enoch** 和组 **root**，则 **chown enoch.root filename** 的结果是 **filename** 归用户 **enoch.root** 所有。您可能是在尝试将文件所有权设置为用户 **enoch** 和组 **root**。这可能会造成混淆。

如果您在同时设置用户和组时始终使用 **chown** 冒号语法，其结果总是能够轻松预测。

默认权限和文件访问

特殊权限对文件和目录的影响

特殊权限	对文件的影响	对目录的影响
u+s (suid)	以拥有文件的用户身份，而不是以运行文件的用户身份执行文件。	无影响。
g+s (sgid)	以拥有文件的组身份执行文件。	在目录中最新创建的文件将其组所有者设置为与目录的组所有者相匹配。
o+t (sticky)	无影响。	对目录具有写入访问权限的用户仅可以删除其所拥有的文件，而无法删除或强制保存到其他用户所拥有的文件。

对可执行文件的 setuid 权限表示将以拥有该文件的用户的身份运行命令，而不是以运行命令的用户身份。以 **passwd** 命令为例：

```
[user@host ~]$ ls -l /usr/bin/passwd  
-rwsr-xr-x. 1 root root 35504 Jul 16 2010 /usr/bin/passwd
```

在长列表中，您可以通过小写的 **s** 辨别出 setuid 权限，该处通常是 **x**（所有者执行权限）。如果所有者不具有执行权限，这将由大写的 **S** 取代。

对某目录的特殊权限 setgid 表示在该目录中创建的文件将继承该目录的组所有权，而不是继承自建用户。这通常用于组协作目录，将文件从默认的专有组自动更改为共享组，或者当目录中的文件始终都应由特定的组所有者使用。**/run/log/journal** 目录就是这样的一个示例：

```
[user@host ~]$ ls -ld /run/log/journal  
drwxr-sr-x. 3 root systemd-journal 60 May 18 09:15 /run/log/journal
```

在长列表中，您可以通过小写的 **s** 辨别出 setgid 权限，该处通常是 **x**（组执行权限）。如果组不具有执行权限，这将会由大写 **S** 取代。

最后，针对目录的粘滞位将对文件删除设置特殊限制。只有文件的所有者（及 root）才能删除该目录中的文件。例如，**/tmp**：

```
[user@host ~]$ ls -ld /tmp  
drwxrwxrwt. 39 root root 4096 Feb 8 20:52 /tmp
```

在长列表中，您可以通过小写的 **t** 辨别出 sticky 权限，该处通常是 **x**（其他执行权限）。如果其他不具有执行权限，这将会由大写 **T** 取代。

umask

用户的默认 umask 由 shell 启动脚本设置。默认情况下，如果您帐户的 UID 为 200 或以上，并且您的用户名和主要组名相同，就会向您分配一个值为 002 的 umask。否则，**您的 umask 将为 022**。

作为 **root** 用户，您可以通过添加名为 **/etc/profile.d/local-umask.sh** 的 shell 启动脚本来更改此设置，该 shell 启动脚本的输出如下例所示：

```
[root@host ~]# cat /etc/profile.d/local-umask.sh
# Overrides default umask configuration
if [ $UID -gt 199 ] && [ "`id -gn`" = "`id -un`" ]; then
    umask 007
else
    umask 022
fi
```

对于 UID 大于 199 且用户名和主要组名相匹配的用户，上面的示例会将 umask 设为 007，其他人的则设为 022。如果只想将每个人的 umask 都设为 022，可以仅使用以下内容创建该文件：

```
# Overrides default umask configuration
umask 022
```

要确保全局 umask 更改生效，您必须注销 shell 并重新登录。在此之前，当前 shell 中配置的 umask 仍然有效。

- 不带参数运行 **umask** 命令将显示 shell 的当前 umask 值。系统上的每个进程都有一个 umask。Bash 的默认 umask 在 **/etc/profile** 和 **/etc/bashrc** 文件中定义。

8. 监控和管理Linux进程

进程的定义

进程是已启动的可执行程序的运行中实例。进程由以下组成部分：

- 已分配内存的地址空间
- 安全属性，包括所有权凭据和特权
- 程序代码的一个或多个执行线程
- 进程状态

进程的环境包括：

- 本地和全局变量
- 当前调度上下文
- 分配的系统资源，如文件描述符和网络端口

现有的（父）进程复制自己的地址空间 (**fork**) 来创建一个新的（子）进程结构。每个新进程分配有一个唯一进程 ID (PID)，满足跟踪和安全性之需。PID 和父进程 ID (PPID) 是新进程环境的元素。任何进程可创建子进程。**所有进程都是第一个系统进程的后代，在红帽企业 Linux 8 系统上，第一个系统进程是 `systemd`。**

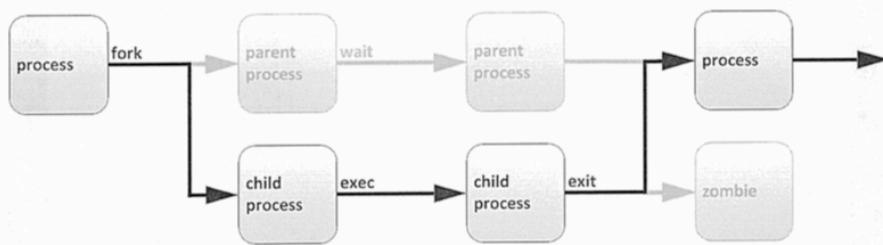


图 8.1: 进程生命周期

通过 `fork` 例程，子进程继承安全性身份、过去和当前的文件描述符、端口和资源特权、环境变量，以及程序代码。随后，子进程可能 `exec` 其自己的程序代码。通常，父进程在子进程运行期间处于睡眠状态，设置一个在子进程完成时发出信号的请求 (`wait`)。在退出时，子进程已经关闭或丢弃了其资源和环境。唯一剩下的资源称为僵停，是进程表中的一个条目。父进程在子进程退出时收到信号而被唤醒，清理子条目的进程表，由此释放子进程的最后一个资源。然后，父进程继续执行自己的程序代码。

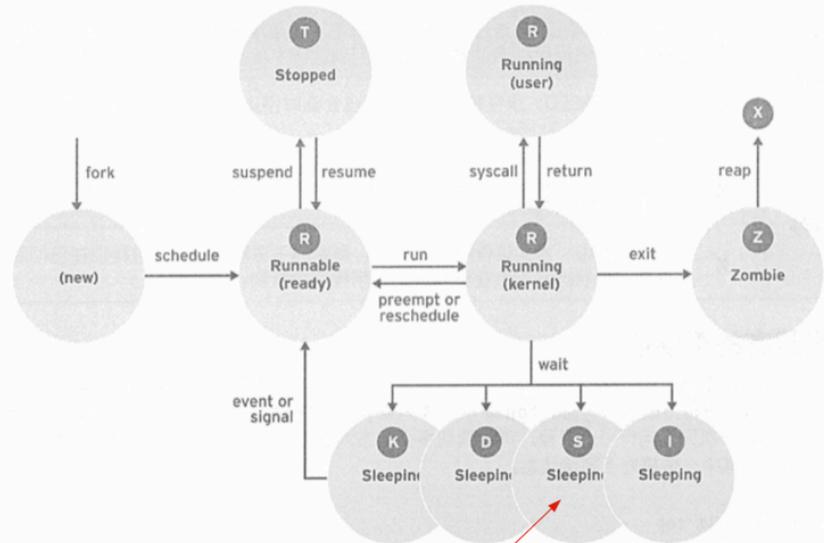


图 8.2: Linux 进程状态

Linux 进程状态在上图中进行了演示，下表也提供了说明。

Linux 进程状态

名称	标志	内核定义的状态名称和描述
运行	红	TASK_RUNNING: 进程正在 CPU 上执行，或者正在等待运行。处于运行中（或可运行）状态时，进程可能正在执行用户例程或内核例程（系统调用），或者已排队并就绪。
睡眠	S	TASK_INTERRUPTIBLE: 进程正在等待某一条件：硬件请求、系统资源访问或信号。当事件或信号满足该条件时，该进程将返回到运行中。
	D	TASK_UNINTERRUPTIBLE: 此进程也在睡眠，但与 S 状态不同，不会响应信号。仅在进程中断可能会导致意外设备状态的情况下使用。
	K	TASK_KILLABLE: 与不可中断的 D 状态相同，但有所修改，允许等待中的任务响应要被中断（彻底退出）的信号。实用程序通常将可中断的进程显示为 D 状态。

名称	标志	内核定义的状态名称和描述
	I	TASK_REPORT_IDLE: D 状态的一个子集。在计算负载平均值时，内核不会统计这些进程。用于内核线程。设置了 TASK_UNINTERRUPTABLE 和 TASK_NOLOAD 标志。类似于 TASK_KILLABLE，也是 D 状态的一个子集。它接受致命信号。
已停止	T	TASK_STOPPED: 进程已被停止（暂停），通常是通过用户或其他进程发出的信号。进程可以通过另一信号返回到运行中状态，继续执行（恢复）。
	T	TASK_TRACED: 正在被调试的进程也会临时停止，并且共享同一个 T 状态标志。
僵停	Z	EXIT_ZOMBIE: 子进程在退出时向父进程发出信号。除进程身份 (PID) 之外的所有资源都已释放。
	X	EXIT_DEAD: 当父进程清理（获取）剩余的子进程结构时，进程现在已彻底释放。此状态从不会在进程列出实用程序中看到。

进程状态的重要性

在对系统进行故障排除时，了解内核如何与进程通信以及进程如何相互通信非常重要。在创建进程时，系统会为进程分配一个状态。**top** 命令的 S 列或 **ps** 的 STAT 列显示每个进程的状态。在单 CPU 系统上，一次只能运行一个进程。可以看到多个状态为 R 的进程。然而，并非所有这些进程都在持续运行，其中一些将处于等待状态。

```
[user@host ~]$ top
PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
1 root 20 0 244344 13684 9024 S 0.0 0.7 0:02.46 systemd
2 root 20 0 0 0 0 S 0.0 0.0 0:00.00 kthreadd
...output omitted...
```

```
[user@host ~]$ ps aux
USER PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMAND
...output omitted...
root 2 0.0 0.0 0 0 ? S 11:57 0:00 [kthreadd]
student 3448 0.0 0.2 266904 3836 pts/0 R+ 18:07 0:00 ps aux
...output omitted...
```

可以使用信号暂停、停止、恢复、终止和中断进程。本章稍后将更为详细地探讨信号。信号可以由其他进程、内核本身或登录系统的用户使用。

列出进程

ps 命令用于列出当前的进程。它可以提供详细的进程信息，包括：

- 用户识别符 (UID)，它确定进程的特权
- 唯一进程识别符 (PID)
- CPU 和已经花费的实时时间
- 进程在各种位置上分配的内存数量
- 进程 **stdout** 的位置，称为控制终端

aux 或许是最常见的选项集，它可显示包括无控制终端的进程在内的所有进程。长列表（选项 **-lax**）提供更多技术详细信息，但可通过避免查询用户名来加快显示。相似的 UNIX 语法使用选项 **-ef** 来显示所有进程。

```
[user@host ~]$ ps aux
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START  TIME COMMAND
root        1  0.1  0.1  51648  7504 ?          Ss  17:45  0:03 /usr/lib/systemd/
systemd
root        2  0.0  0.0      0     0 ?          S   17:45  0:00 [kthreadd]
root        3  0.0  0.0      0     0 ?          S   17:45  0:00 [ksoftirqd/0]
root        5  0.0  0.0      0     0 ?          S< 17:45  0:00 [kworker/0:0H]
root        7  0.0  0.0      0     0 ?          S   17:45  0:00 [migration/0]
...output omitted...
[user@host ~]$ ps lax
F   UID      PID  PPID PRI  NI    VSZ   RSS WCHAN  STAT TTY      TIME COMMAND
4   0        1      0  20   0  51648  7504 ep_pol Ss  ?      0:03 /usr/lib/
systemd/
1   0        2      0  20   0      0  kthrea S   ?      0:00 [kthreadd]
1   0        3      2  20   0      0  smpboo S   ?      0:00 [ksoftirqd/0]
1   0        5      2  0 -20   0      0  worker S<  ?      0:00 [kworker/0:0H]
1   0        7      2 -100  -      0  smpboo S   ?      0:00 [migration/0]
...output omitted...
[user@host ~]$ ps -ef
UID      PID  PPID  C STIME TTY      TIME CMD
root        1      0  0 17:45 ?      00:00:03 /usr/lib/systemd/systemd --
switched-ro
root        2      0  0 17:45 ?      00:00:00 [kthreadd]
root        3      2  0 17:45 ?      00:00:00 [ksoftirqd/0]
root        5      2  0 17:45 ?      00:00:00 [kworker/0:0H]
root        7      2  0 17:45 ?      00:00:00 [migration/0]
...output omitted...
```

默认情况下，如果不使用选项而运行 **ps**，将选择具有与当前用户相同的有效用户 ID (EUID) 并与调用 **ps** 所处同一终端关联的所有进程。

· 方括号中的进程（通常位于列表顶部）为调度的内核线程。

· 僵停列为 **exiting** 或 **defunct**。

· **ps** 的输出显示一次。使用 **top** 来获得

地址：广州市天河区棠安路 188 号乐天大厦二楼

在后台运行作业

任何命令或管道都可以在后台启动，只需在命令行的结尾处附加符号 & 即可。Bash shell 显示作业编号（特定于会话的唯一编号）和新建子进程的 PID。shell 不等待子进程终止，而会显示 shell 提示符。

```
[user@host ~]$ sleep 10000 &
[1] 5947
[user@host ~]$
```



注意

如果利用 & 符号将包含管道的命令行发送到后台，管道中最后一个命令的 PID 将用作输出。管道中的所有进程仍是该作业的成员。

```
[user@host ~]$ example_command | sort | mail -s "Sort output" &
[1] 5998
```

您可以使用 **jobs** 命令显示 Bash 为特定会话跟踪的作业列表。

地址：广州市天河区棠安路 188 号乐天大厦二楼

电话：020-38289118

网址：www.togogo.net

RH124-L8.0-zh_cn-1-20190531



```
[user@host ~]$ jobs
[1]+  Running                  sleep 10000 &
```

可以使用 **fg** 命令和后台作业的 ID（%作业编号）将该后台作业转至前台。

```
[user@host ~]$ fg %1
sleep 10000
```

在上例中，**sleep** 命令现在正在控制终端的前台中运行。shell 自身再次睡眠，等待这一子进程退出。

如要将前台进程发送到后台，请首先在终端中按键盘生成的暂停请求 (**Ctrl+z**)。

```
sleep 10000
^Z
[1]+  Stopped                 sleep 1
```

ps j 命令显示与作业相关的信息。PID 是进程的唯一进程 ID。PPID 是此进程的父进程（即启动（分叉）此进程的进程）的 PID。PGID 是进程组首进程的 PID，通常是作业管道中的第一个进程。SID 是会话首进程的 PID，对于作业而言，这通常是正在其控制终端上运行的交互 shell。由于示例 **sleep** 命令当前已暂停，因此其进程状态为 **T**。

```
[user@host ~]$ ps j
PPID  PID  PGID  SID TTY      TPGID STAT   UID    TIME COMMAND
2764  2768  2768  2768 pts/0    6377 S+    1000   0:00 /bin/bash
2768  5947  5947  2768 pts/0    6377 T     1000   0:00 sleep 10000
2768  6377  6377  2768 pts/0    6377 R+    1000   0:00 ps j
```

要启动在后台运行的已暂停进程，可使用 **bg** 命令及相同的作业 ID。

```
[user@host ~]$ bg %1
[1]+  sleep 10000 &
```

如果用户尝试退出带有暂停作业的终端窗口（会话），那么 shell 将发出警告。如果用户再次尝试立即退出，暂停的作业将被中断。

中断进程

使用信号控制进程

信号是传递至进程的软件中断，信号向执行中的程序报告事件。生成信号的事件可以是错误或外部事件（I/O 请求或定时器过期），或者来自于显式使用信号发送命令或键盘序列。

下表列出了系统管理员用于日常进程管理的基本信号。请通过短名称 (HUP) 或正确名称 (SIGHUP) 指代信号。

基本进程管理信号

信号 编号	短名称	定义	用途
1	HUP	挂起	用于报告终端控制进程的终止。也用于请求进程重新初始化（重新加载配置）而不终止。
2	INT	键盘中断	导致程序终止。可以被拦截或处理。通过按 INTR 键序列 (Ctrl+c) 发送。
3	QUIT	键盘退出	与 SIGINT 相似；在终止时添加进程转储。通过按 QUIT 键序列 (Ctrl+\) 发送。
9	KILL	中断，无法拦截	导致立即终止程序。无法被拦截、忽略或处理； 总是致命的 。
15 默认	TERM	终止	导致程序终止。和 SIGKILL 不同，可以被拦截、忽略或处理。要求程序终止的“友好”方式；允许自我清理。
18	CONT	继续	发送至进程使其恢复（若已停止）。无法被拦截。即使被处理，也始终恢复进程。
19	STOP	停止，无法拦截	暂停进程。无法被拦截或处理。
20	TSTP	键盘停止	和 SIGSTOP 不同，可以被拦截、忽略或处理。通过按 SUSP 键序列 (Ctrl+z) 发送。

每个信号都有一个默认操作，通常是如下之一：

- 终止 — 导致程序立即终止（退出）。
- 核心转储 — 导致程序保存内存镜像（核心转储），然后终止。
- 停止 — 导致程序停止执行（暂停），再等待继续（恢复）。

程序可以通过实施处理程序例程来为响应预期的事件信号做准备，以忽略、替换或扩展信号的默认操作。

通过明确请求发送信号的命令

您可以向当前的前台进程发送信号，具体操作为按下键盘控制序列以暂停 (**Ctrl+z**)、中止 (**Ctrl+c**) 或核心转储 (**Ctrl+**) 该进程。不过，您将使用信号发送命令向后台进程或另一会话中的进程发送信号。

可以通过名称（例如 **-HUP** 或 **-SIGHUP**）或编号（相关的 **-1**）将信号指定为选项。用户可以中断自己的进程，但需要 root 特权才能终止由其他人拥有的进程。

kill 命令根据 PID 编号向进程发送信号。虽然其名称为 **kill**，但该命令可用于发送任何信号，而不仅仅是终止程序的信号。您可以使用 **kill -l** 命令列出所有可用信号的名称和编号。

```
[user@host ~]$ kill -l
 1) SIGHUP      2) SIGINT      3) SIGQUIT      4) SIGILL      5) SIGTRAP
 6) SIGABRT     7) SIGBUS      8) SIGFPE       9) SIGKILL     10) SIGUSR1
 11) SIGSEGV    12) SIGUSR2    13) SIGPIPE     14) SIGALRM    15) SIGTERM
 16) SIGSTKFLT   17) SIGCHLD    18) SIGCONT     19) SIGSTOP    20) SIGTSTP
...output omitted...
[user@host ~]$ ps aux | grep job
5194  0.0  0.1 222448  2980 pts/1    S    16:39   0:00 /bin/bash /home/user/bin/
control job1
5199  0.0  0.1 222448  3132 pts/1    S    16:39   0:00 /bin/bash /home/user/bin/
control job2
5205  0.0  0.1 222448  3124 pts/1    S    16:39   0:00 /bin/bash /home/user/bin/
control job3
5430  0.0  0.0 221860  1096 pts/1    S+   16:41   0:00 grep --color=auto job
[user@host ~]$ kill 5194
[user@host ~]$ ps aux | grep job
user  5199  0.0  0.1 222448  3132 pts/1    S    16:39   0:00 /bin/bash /home/
user/bin/control job2
user  5205  0.0  0.1 222448  3124 pts/1    S    16:39   0:00 /bin/bash /home/
user/bin/control job3
user  5783  0.0  0.0 221860   964 pts/1    S+   16:43   0:00 grep --color=auto
job
[1]  Terminated                  control job1
[user@host ~]$ kill -9 5199
[user@host ~]$ ps aux | grep job
user  5205  0.0  0.1 222448  3124 pts/1    S    16:39   0:00 /bin/bash /home/
user/bin/control job3
```

地址：广州市天河区棠安路 188 号乐天大厦二楼

killall 命令可以根据命令名称向多个进程发送信号。

```
[user@host ~]$ ps aux | grep job
5194 0.0 0.1 222448 2980 pts/1 S 16:39 0:00 /bin/bash /home/user/bin/
control job1
5199 0.0 0.1 222448 3132 pts/1 S 16:39 0:00 /bin/bash /home/user/bin/
control job2
5205 0.0 0.1 222448 3124 pts/1 S 16:39 0:00 /bin/bash /home/user/bin/
control job3
5430 0.0 0.0 221860 1096 pts/1 S+ 16:41 0:00 grep --color=auto job
[user@host ~]$ killall control
[1] Terminated control job1
[2]- Terminated control job2
[3]+ Terminated control job3
[user@host ~]$
```

使用 **pkill** 向一个或多个符合选择条件的进程发送信号。选择条件可以是命令名称、特定用户拥有的进程，或所有系统范围进程。**pkill** 命令包括高级选择条件。

- 命令 — 具有模式匹配的命令名称的进程。
- UID — 由某一 Linux 用户帐户拥有的进程，无论是有效的还是真实的。
- GID — 由某一 Linux 组帐户拥有的进程，无论是有效的还是真实的。
- 父级 — 特定父进程的子进程。
- 终端 — 运行于特定控制终端的进程。

```
[user@host ~]$ ps aux | grep pkill
user 5992 0.0 0.1 222448 3040 pts/1 S 16:59 0:00 /bin/bash /home/
user/bin/control pkill1
user 5996 0.0 0.1 222448 3048 pts/1 S 16:59 0:00 /bin/bash /home/
user/bin/control pkill2
user 6004 0.0 0.1 222448 3048 pts/1 S 16:59 0:00 /bin/bash /home/
user/bin/control pkill3
[user@host ~]$ pkill control
[1] Terminated control pkill1
[2]- Terminated control pkill2
[user@host ~]$ ps aux | grep pkill
user 6219 0.0 0.0 221860 1052 pts/1 S+ 17:00 0:00 grep --color=auto
pkill
[3]+ Terminated control pkill3
[user@host ~]$ ps aux | grep test
user 6281 0.0 0.1 222448 3012 pts/1 S 17:04 0:00 /bin/bash /home/
user/bin/control test1
user 6285 0.0 0.1 222448 3128 pts/1 S 17:04 0:00 /bin/bash /home/
user/bin/control test2
user 6292 0.0 0.1 222448 3064 pts/1 S 17:04 0:00 /bin/bash /home/
user/bin/control test3
```

地址：广州市天河区棠安路 188 号乐天大厦二楼

电话：020-38289118

以管理员身份注销用户

出于各种各样的原因，您可能需要注销其他用户。以下是许多可能性中的一些示例：用户做出了安全违规行为；用户可能过度使用了资源；用户的系统可能不响应；或者，用户不当访问了资料。在这些情况下，您可能需要以管理员身份使用信号来终止其会话。

要注销某个用户，首先确定要终止的登录会话。使用 **w** 命令列出用户登录和当前运行的进程。记录 **TTY** 和 **FROM** 列，以确定要关闭的会话。

所有用户登录会话都与某个终端设备 (TTY) 相关联。如果设备名称的格式为 **pts/N**，说明这是一个与图形终端窗口或远程登录会话相关联的伪终端。如果格式为 **ttyN**，则说明用户位于一个系统控制台、替代控制台或其他直接连接的终端设备上。

```
[user@host ~]$ w
12:43:06 up 27 min 5 users, load average: 0.03, 0.17, 0.66
USER   TTY      FROM          LOGIN@    IDLE     JCPU   PCPU WHAT
root   tty2          12:26 14:58  0.04s  0.04s -bash
bob    tty3          12:28 14:42  0.02s  0.02s -bash
user   pts/1  desk.example.com 12:41  2.00s  0.03s  0.03s w
[user@host ~]$
```

查看会话登录时间，了解用户已登录该系统的时长。对于每个会话，当前作业耗用的 CPU 资源（包括后台任务和子进程）位于 **JCPU** 列中。当前的前台进程 CPU 占用情况列在 **PCPU** 列中。

可以逐一向进程和会话发送信号，也可以集体发送。要终止一个用户的所有进程，可使用 **pkill** 命令。由于登录会话中的初始进程（会话首进程）设计为可以处理会话终止请求并忽略不需要的键盘信号，中断某一用户的所有进程和登录 shell 需要使用 SIGKILL 信号。



重要

管理员通常太快使用 SIGKILL。

由于 SIGKILL 信号无法被处理或忽略，它总是致命的。然而，它会强行终止进程，而不允许被中断的进程运行自我清理例程。建议先发送 SIGTERM，然后尝试 SIGINT；只有在这两个进程都失败时通过 SIGKILL 重试。

首先使用 **pgrep** 确定要中断的 PID 编号，此操作与 **pkill** 非常相似，包括使用相同的选项，但 **pgrep** 是列出进程而非中断进程。